Canadian Journal of Learning and Technology
Volume 33(1) Winter / hiver 2007
Wicked ID: Conceptual Framework for Considering Instructional Design as a Wicked Problem

Katrin Becker

Auteur

Katrin Becker, Serious Games Research, Graduate Division of Educational Research, University of Calgary. Correspondence regarding this article can be sent to: beckerk@ucalgary.ca or becker@minkhollow.com

Abstract

Abstract: The process of instructional design has parallels in other design disciplines. Software design is one that has experienced intense attention in the last 30 or so years, and many lessons learned there can be applied to ID. Using software design as a springboard, this concept paper seeks to propose a new approach to ID. It suggests that instructional design is almost always a Wicked Problem. The connection is formed between Wicked Problems as first described by Rittel and Webber in 1973, and the models in and processes of instructional design. The areas of social planning, organizational management and software design all posses some accepted and tested approaches to the solution of Wicked Problems. These will be described, and how they can be applied to ID will be explained. Finally, this paper will propose a meta-model for ID and explain how it can be used in the current context.

Résumé : Le processus de conception pédagogique a des similitudes avec d'autres disciplines. La conception de logiciels a fait l'objet d'une grande attention dans les trente dernières années et de nombreuses leçons tirées peuvent s'appliquer à la conception pédagogique. En se basant sur la conception de logiciels, cet article cherche à proposer une nouvelle méthode à la conception pédagogique. Il laisse entendre que la conception pédagogique constitue presque toujours un problème épineux. Le lien est formé entre les problèmes épineux tels que décrits par Rittel et Webber en 1973 et les modèles et processus de la conception pédagogique. Les secteurs de l'organisation sociale, de la gestion des organisations et de la conception de logiciels comptent des méthodes acceptées et validées à la résolution de problèmes épineux. Nous les décrirons et expliquerons comment ils peuvent servir à la conception pédagogique. Enfin, l'article proposera un meta-modèle de la conception pédagogique et expliquera comment il peut servir dans le contexte actuel.

Introduction

Design is a complex activity. Each design discipline has its own domain, but all share at least some common attributes and challenges. Instructional design has some parallels with software design, not the least of which is that both offer many different models for creating detailed specifications for the development, implementation, evaluation, and maintenance of their respective 'products' (that being software and instructional interventions). While both make at least some claim to a status as a "mature profession", as defined by Ford and Gibbs (1996), these elements are not well entrenched in either. Ford and Gibbs identified eight elements of a mature profession, most of which are present to a greater or lesser extent in both instructional and software design. These elements are:

1. Initial professional education (graduate and undergraduate curricula)
2. Accreditation of university programs
3. Skills development (co-op programs, internships, apprenticeships, etc.)
4. Certification of practitioners
5. Licensing of practitioners
6. Professional development (additional study undertaken after professional practice)
7. Code of ethics
8. Professional societies

Mature status notwithstanding, new design models continue to evolve in both disciplines as significant issues remain. In software design, "extreme programming" (XP) (Beck, 2000) , "agile methods" (Krill, 2006) , and "lean software" (Poppendieck & Poppendieck, 2003) have all enjoyed considerable popularity. Even though most of the concepts are far from new, the processes have been given a 'new coat of paint' and formal names, and so are treated as new ideas. The concept of extreme programming (aka 'XP') includes a dozen rules intended to prescribe the work process, such as, "The system is put into production in a few months, before solving the whole problem." (Beck, 1999, p. 71) Many of the "rules" are familiar to virtually all programmers over 40 as they are the same practices that have been employed by experts for decades. The agile methods proponents have a manifesto that includes valuing "Individuals and interactions over processes and tools", "Working software over comprehensive documentation", "Customer collaboration over contract negotiation", and "Responding to change over following a plan" (Beck, Beedle et al., 2001) While the goals are certainly laudable, the ideas predate notions of software engineering as a profession (Weinberg, 1998) , and attaining them has been difficult (Krill, 2006) . Similarly, many of the practices advocated by the "lean software" movement were described over 30 years ago (Brooks, 1975) . Aspect-Oriented Programming (AOP), a relatively new perspective (Kiczales et al., 1997) , is still gaining an audience. The basic idea is that there are various aspects, or concerns (such as dealing with invalid inputs) that span multiple parts of a software system. Typical object-oriented approaches result in considerable duplication of code as these "concerns" are dealt with in various modules.

Whether or not the 'new' methods are truly new or not, a question that begs to be asked is, "Why is there still so much active development of process models in what claims to be a mature profession?" In spite of spending considerable effort on the problem, no single,

right model has yet emerged. The reason for this is simple: the right one doesn't exist because there is no right one. The general problem of finding a model that can be used to support the design process remains unsolved.

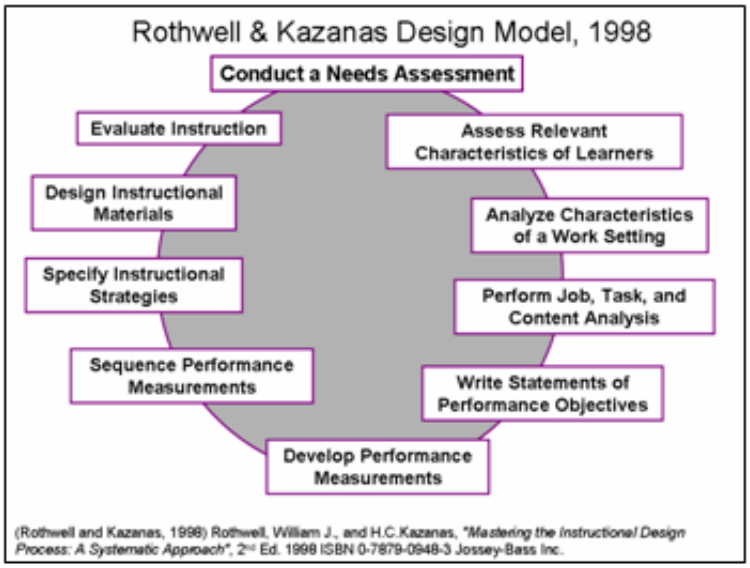
Design is a creative process that requires healthy doses of innovation, and, according to Michael Scharge (2000), "The most important raw material of innovation has always been the interplay between individuals and the expression of their ideas" (p. 13) . And yet, we keep trying to create prescriptive models. Various researchers have taken different approaches in their efforts to address the design problem. Some feel we need to perfect the models we have, and become more skilled at applying them—software engineering is in large part devoted to this approach. However, the reality seems to be that in spite of the best efforts of many expert minds we are no closer to an "ideal" software design model than we were 30 years ago. Even worse, there is still no conclusive evidence that the methods thus far developed actually contribute to safer, or more error-free, or easier to maintain systems in spite of claims to the contrary, such as, "Lean Software Development helps you refocus development on value, flow and people—so you can achieve breakthrough quality, savings, speed, and business alignment" (Poppendieck & Poppendieck, 2003, back cover).

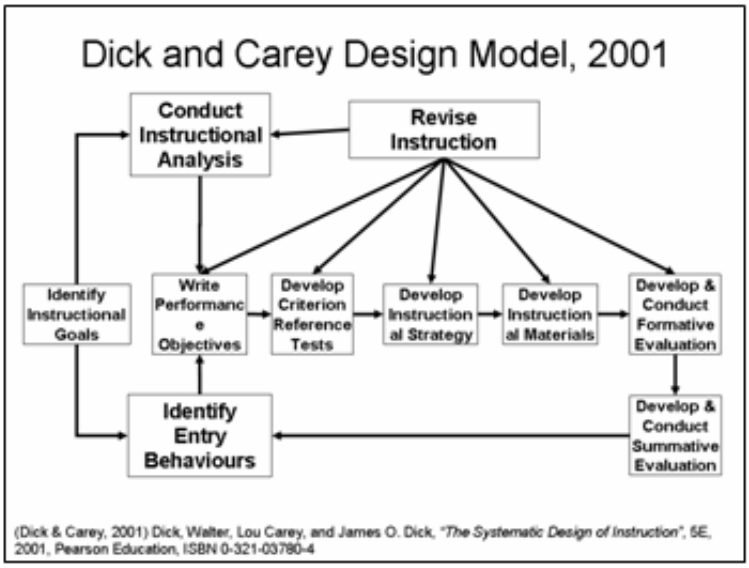
The Current State of Instructional Design Models

Much the same can be said of instructional design models and processes, both in theory and in practice: there exist many models, with new ones always being developed. Some experts devote their efforts to perfecting the models they have already developed, such as Shank and Merrill, while others (Brophy & Velankar, 2006; Morrison, Ross, & Kemp, 2004; Willis, 2000) are developing new models. As in software engineering (SENG), instructional design (ID) also entertains considerable debate over the applicability and efficacy of the models that exist. In some ways, however, ID is already more erudite than SENG. In ID there seems to be a general recognition that no one method can work in all situations. And even those who support the most structured approaches will admit that these are often best suited to use by practitioners new to the field, by providing a support system. Experts who make use of these models often use them as rough guides, rather than prescriptions (Kenny, Zhang, Schwier, & Campbell, 2005).

Although a thorough treatment of the current state of ID is beyond the scope of this discussion, it is useful to provide a brief overview to allow the proposed perspective to be located in the existing landscape. Below is a sampling of six well-known ID models (Dick, Carey, & Carey, 2001; Gerlach & Ely, 1980; Hannafin & Peck, 1988; Rothwell & Kazanas, 1998; Tripp & Bichelmeyer, 1990; Wiggins & McTighe, 1998). Each of the six models contains valuable insights and guidelines for development of instruction. Each addresses various problems effectively, and it would be foolish to ignore or discard them in an attempt to create yet another detailed model. This is not the goal of this paper; rather, this paper proposes to incorporate those elements common to all in a manner that befits the ill-defined nature of many instructional situations. Regardless of which discipline is examined, one finds successful creators who attribute much of their success to their ability

to use effectively the insights of those who came before. It is unlikely that ID is unique in this respect, and this paper seeks to build on those developments rather than to supplant them.





When we look specifically at these ID Models, we notice that most include many of the same specific elements—in most cases the primary differences have to do with the way sub-parts are categorized and the placement of the 'bubbles' and connectors in the visual representations. While some differences may appear to be major from some perspectives, especially those of a novice, a systems analyst's view is that the actual difference is one of sorting (organization), rather than content. Looking beyond the visual presentation and descriptive terminology, we find that most are firmly anchored to the instructional technology domains of design, development, utilization, management, and evaluation as defined by Seels and Richey (1994). This kind of similarity of modes also tends to hold true for the set of design models found in software design (Budgen, 2003), as well as other domains. It appears that all design models, regardless of domain (software, architecture, fashion, etc.) contain the following five elements in one form or another.




General Design Template

Requirements Analysis: identifying ***what*** is needed from a solution

Specification: a description of what the solution must ***do*** to meet the requirements

Design: describes ***how*** the solution will do what it must do

Implementation: elaborates on the design and translates it into a form that can be ***used***

Testing: validation; demonstrates how well it does what it is supposed to do

Many instructional design models have similar elements and the well-known ADDIE template that often forms the basis for these models (Molenda, 2003) differs from the above general one primarily in how the tasks are divided and organized rather than the nature of the tasks. This difference highlights the pervasive social context that is part of all instructional endeavours. The ADDIE model consists of five elements (analysis, design, development, implementation, evaluation) which combine the analysis and specification elements into a single analysis element, while breaking up the design and implementation elements into three parts with the addition of a development element between the two others. Each of the models shown in this paper has been carefully thought out—each is complete and useable. When put to practice, however, the application of the model is rarely as clean as one hopes—each phase remains subject to influence from other phases —often right up until the completion of the project or even its ultimate delivery. While it may be true that experts tend to use these models as theoretical frameworks, those closer to the novice end of the spectrum look to these models as process guides (Kenny et al., 2005). Others have suggested that this model is in fact more of a conceptual framework than a process model (Bichelmeyer, Boling, & Gibbons, 2006).

More recently, some have begun to look beyond the structured and engineered approaches to the design of instruction. There is growing concern over the lack of focus on the enterprise of instructional design (Bichelmeyer et al., 2006) at the same time as others advocate that instructional design should be emulating software engineering processes even more (Douglas, 2006). Lack of solid evidence for the effectiveness of software engineering methods aside (Wang, 2002), one question that needs to be considered carefully is whether we really want to view instruction and learning as 'products'. The result of any software engineering effort may be comfortably referred to as a 'product' but doing the same for the end-result of an instructional intervention has implications that are antithetical to prevailing learner-centered perspectives. Indeed some current instructional designers do not espouse the engineering metaphor and for them the earlier clarity of ISD has given way to a far cloudier field of view (Schwier, Hill, Wager, & Spector, 2006). This

new 'cloudy' view although far less clear than the views of the previous decades has seen a major shift of perspective from the design of instruction to the design of environments and situations that foster learning (Jonassen, Cernusca, & Ionas, 2006). Among these are Jonassen's application of Activity Theory to the design of constructivist learning environments (Jonassen & Rohrer-Murphy, 1999), cultural and collective intelligences in Varela's Theory of Enactivism (Varela, Thompson, & Rosch, 1991), and Andrew Gibbons' view of design as a series of layers (media-centrism, message-centrism, strategy-centrism, and model-centrism) that change and evolve as the designer gains experience (Gibbons, 2003). While these developments are very encouraging for the future of ID, most are descriptive rather than prescriptive, and this leaves many novices with little support when it comes to learning how to become practitioners in their field. There is clearly room for further development and insight. One explanation for the lack of clarity in ID is that ID falls into that category of problem known as "Wicked".

Ill-Structured vs. Wicked Problems

David Jonassen has provided us with valuable insights into approaches to the instructional design of problem solving (Jonassen, 1997, 2004), but the meta-view of ID requires us to take things to the next level and discuss instructional design as its own kind of problem. Most problems of instructional design, and for that matter, many other design activities have many of the properties of what are now widely known as Ill-Structured. Ill-Structured Problems (ISP) were originally described by Herbert Simon in 1973 (Kenny et al., 2005), and their properties include: a lack of agreement on solutions and solution paths, as well as a high degree of uncertainty about answers to problems. These are problems for which the existing and desired states are unclear and the method of getting to the desired state is unknown.

Around the same time as Simon's work on ill-structured problems (Simon, 1973) became known (Derry & Steinkuehler, 2003; Donald, 2001; Lakoff, 1987; Stewart & Cohen, 1997; Tomasello, 1999), Rittel and Webber published their seminal paper describing ten criteria for another class of problem, which they labelled "Wicked" (Rittel & Webber, 1973). Wicked Problems have some of the same properties as ill-structured problems (ISP), but also have some additional criteria to further complicate the solution process. The term "Wicked Problem" was first coined in the context of social planning. Originally, it had little connection to other disciplines, but perhaps due to its social context, it was soon taken up by business organizations and applied to organizational planning, as well as in architectural contexts. The idea also spread into software engineering but only some parts remained popular, such as the "scrum" approach to the design process, to be described later in this paper. Whatever the reason, wicked problems have not received the same level of exposure, acceptance and application as ill-structured problems have. If one examines both definitions, however, it quickly becomes apparent that the process of instructional design is often better described as a wicked problem than an ill-structured one. The social context of the problem, which is central to the wicked problem concept, is in this case pivotal. It is the direct influence from and on the people involved in the solution that

makes social planning a wicked problem, and thus, especially in constructivist contexts, instructional design, having a similar social context, is also a wicked problem.

Wicked Problems

Problems rarely fall neatly into nice, clean categories, but Rittel and Webber (DeGrace & Stahl, 1990) distinguish two ends of a continuum: Wicked Problems exist at one end, and those they refer to as 'tame' at the other. Now, 'tame' problems are not necessarily easy. Instead, what distinguishes them from the wicked kind is that a tame problem tends to be well understood; it can be solved using sequential processes (like those typically shown in flowchart style visualizations); it begins with an unambiguous problem statement, and, finally, ends with a correct (read: verifiable) solution. Many forms of classical training, especially those involving physical skills, can be considered tame, even though it would be naive to label them easy. Often attempts are made to address wicked problems by recasting them as tame ones. "By treating a wicked problem as a tame problem, energy and resources are misdirected, resulting in solutions that not are only ineffective, but can actually create more difficulty" (Nelson & Stolterman, 2003, 1973).

The following list outlines the ten criteria identified as being associated with Wicked Problems as defined by Rittel and Webber (1973). Following the list is an explanation of each and how it can be related to instructional design.

10 main criteria for wicked problems

1. There is no definitive formulation of a Wicked Problem.
2. Wicked Problems have no stopping rule[1].
3. Solutions are not True/False but Good/Bad.
4. There is no ultimate test of a solution to a Wicked Problem.
5. Each solution is a one shot operation.
6. Wicked Problems do not have enumerable (exhaustively describable) solutions.
7. Each problem is unique.
8. Each problem is a symptom of another problem.
9. There are a number of different stakeholders interested in how it is solved.
10. The planner has no right to be wrong. In other words there is a (perhaps) unreasonable expectation that the designers will produce a suitable and sound solution in the first attempt.

How ID is Wicked

> When a practitioner reflects experientially, three events not found in the theory-to-practice model occur. First, the practitioner approaches problems, not as copies of generalized theory but as unique, personal instances. Here, the practitioner pays attention to the non-confirming or anomalous aspects of a problem – those characterized by *'uncertainty, instability, uniqueness, and value conflict'* (Rittel & Webber, 1984) . The practitioner's art is that of working through this 'mess', not by applying universal rules but by employing intuition, analogies, metaphors. (Schön, 1983, p. 40)

Wicked problems exist in social contexts, where that is defined as situations involving more than one person. ID certainly qualifies here, as even in situations where the learner is far removed from the facilitator in time or space, they are both part of a social group, the intervention is likely to be used more than once, and what is learned is rarely applied in social isolation. By contrast, many software applications do not have a social context. A compiler may be used by thousands or even millions of people and may be a complex piece of software, but it does not have a social context. Its function can be measured quite

objectively and indeed they are often tested in an automated way. Designing an energy efficient home is a difficult and complex design problem, but it too lacks a social context and would not be deemed wicked. Even though people will probably live in the house, the design of the house itself is not contextualized in a social domain. It is possible to define clear and precise measures for the known parameters that define energy efficiency. These measures do not require a social context merely geographical, climactic, and scientific ones which will allow designers to determine when they have completed their task. However, designing an energy efficient home *that people will like*, and that is harmonious with its surroundings *does* place it in a social context, and then it starts to look more like a wicked problem. The first can be assessed using relatively objective measures (such as heat loss under various conditions), but the second adds criteria, such as "liking" and "harmonious", that are challenging if not impossible to measure objectively. This subjectivity is part of what gives a problem its social context.

The following paragraphs outline the connection between wicked problems and instructional design. Note that even if a problem does not meet all the criteria for a wicked problem, it can still be classified as an ill-structured problem, or something in between. If it has too few of the listed criteria, then it is probably best categorized differently. Still, approaches used to solve wicked problems will often be just as effective when the problem is merely ill-structured, provided it falls in a social domain. A key feature that distinguishes problems in the social domain from others is that the problem goal in social contexts typically involves improving some aspect of the environment in which people live and work, rather than the determination of some truth.

1. There is no definitive formulation of a Wicked Problem.

A problem is progressively understood as solutions are developed. In many cases the problem is not really understood until a solution is at hand. If, for example, the identified problem is a lack of formal thinking skills in freshman college students (Doll, 1993), then the solution would appear to be to improve their formal thinking skills, but knowing that doesn't really bring us closer to a solution. Should the intervention, whatever it turns out to be, be applied in high school or college? These may require drastically different approaches. Each step in the specification of the problem is also a step towards a distinct solution, and each stakeholder will have a distinct interpretation of what the problem is. In other words, our understanding of the problem is linked with ideas we may have about solving it (Parham, 2003).

It may be possible to precisely define a wicked problem at a fairly high level, such as: create a single semester course that teaches college-level introductory physics, or identify a performance gap such as freshman college students are unable to solve story-problems. Beyond that, the description of the problem becomes far more subjective. Each person involved in the solution may bring a different perspective.

2. Wicked Problems have no stopping rule.

In mathematics, a "stopping rule" is a condition that indicates the problem has been

solved. The stopping rule for cooking soft-boiled eggs is usually to leave the eggs in boiling water for three minutes (then stop). Since there is no definitive formulation of the problem in a wicked problem that means it is also difficult to know when we have solved it. Progress towards a solution typically continues until we run out of resources (Budgen, 2003), one of which is time. There is no clear point when we can determine we are finished, and there is no intrinsic measure of completion.

Most ID models are cyclic in nature and so already recognize the necessity for a solution to undergo regular evaluation and revision. However, very few indicate that a certain amount of 'drift' is also inherent—meaning that as the revisions progress, the exact nature of the problem that is being 'solved' also changes.

3. Solutions are not True/False but Good/Bad.

There is very little to be gained by searching for "the right answer" to a wicked problem. There isn't one. There may be many "right answers", or there may be no solution at all that can be labelled "right", merely one that is 'good'. Our quest is for a solution that will be 'good enough'. Of course, we have just created a new problem: "How do we define 'good enough'?", but that is another discussion.

This is another aspect of ID that has been acknowledged for some time, at least to a degree. Even in the early days of the discipline, ID models did not generally attempt to create correct solutions, but rather sought to create standards of quality in determining the adequacy of instruction and learning (Conklin & Weil, 1997).

4. There is no ultimate test of a solution to a Wicked Problem.

We often won't fully understand the problem until we have developed a solution. While it may be possible to test our solution for various qualities, such as specific performance improvements, it is unlikely we will ever be able to test it conclusively. Just as the number of paths through the development of a solution is potentially infinite, the testing of a solution may be equally infinite. We are never really done. Testing can essentially only hope to provide us with a measure of relative suitability. Progress toward a solution tends to be defined qualitatively in terms of how much more is understood about the problem rather than as a distance from the solution (Seels & Richey, 1994).

To further complicate matters, in ID the viability of a potential solution can only be postulated (no matter how carefully analyzed or thoroughly evaluated) until it has actually been tried on real people and assessed. Some assessments can determine suitability quite well, but it is also possible that there are as many potential tests of suitability as there are stakeholders.

5. Each solution is a one shot operation.

While portions of a solution may be applicable to other problems, each similar problem will retain sufficient distinct characteristics to render it unique. In the original context of wicked problems, that is, social planning, something as seemingly simple as the placement of a homeless shelter depends on so many other factors that the solution for one

neighbourhood could never be directly applied to another. In computer science, the choice of programming language to be used in the introductory programming course remains a topic of debate. MIT (Massachusetts Institute of Technology), uses a functional language called Scheme, and has found it to be quite effective. This same approach cannot be used in most other schools, as they do not have the same kinds of faculty and students among other things. MIT chooses its students from among the very best so when the faculty can assume students will meet any standards and requirements they set, and most incoming students already have substantial programming experience so their freshmen are not novices. Again, the social context of MIT is unlike that of most other schools, so their solution would not necessarily work anywhere else.

In ID, each instructional solution is significant as soon as it gets used, even if it is used incorrectly or fails to meet the intended outcome. In other words it always has an effect. This is often not true of software solutions. In education, there really is no way to 'try it out' without affecting someone, and human nature being what it is, the recipients of the instruction will almost certainly have learned *something* as a result of the instruction. The limited generalizability of solutions is well documented (Fitzpatrick, 2003), and any teacher with a few years' experience can tell you that a technique that worked well in one class may fail completely in another.

6. Wicked Problems do not have enumerable (exhaustively describable) solutions .

Problems in ID can rarely be solved by choosing one of 'N' solutions, because it is not possible to list all the possible solutions. There is also no well-described set of permissible operations that may be incorporated into the plan. There are effectively an infinite number of possible solutions. Each stakeholder will have a different set of potential solutions, and determining when a sufficient number of potential solutions have emerged is a matter of judgment, rather than the result of testing (Bereiter, 2002; Meltzer, 2004). Indeed, it is possible that *no* solution may be found. "The set of feasible plans of action relies on realistic judgment, the capability to appraise 'exotic' ideas and the amount of trust and credibility." (Rittel & Webber, 1973, p. 164) Often the conclusion to the planning comes with a statement like, "OK, Let's try that."

Suppose we are trying to create an intervention that will teach the concept of recursion to novice computer scientists. This is a problem that has vexed computer science instructors for decades (Poppendieck, 2004), and remains essentially an unsolved problem (Kruse, 1982). More than a quarter century of effort has neither solved the problem, nor exhausted the possible solutions. Many individuals have solutions that they prefer, but there is still no well-described set of possible techniques that are known to lead to an understanding of recursion.

7. Each problem is unique.

There is no well-defined or understood algorithm for proceeding from some given problem to the solution. Each solution is affected by the other elements involved: a change to one of the elements may result in a different problem. Guidelines abound, but creating an

instructional solution cannot be reduced to a recipe that can be blindly followed.

Given two 'identical' classes taught in different years, or different places, they will both have qualities unique to themselves. For example, the design of a university transfer course at a regional college will be a different problem from the design of the 'same' course at the target institution.

8. Each problem is a symptom of another problem.

Resolving one part of the problem may create or uncover another problem in turn. By settling on a specific approach to one portion, we have effectively changed the original problem, and we may well create difficulties in another portion. Wicked problems involve an interlocking set of constraints that change over time. As a result, the problem should be settled on as high a level of abstraction as possible. For example, resolving the choice of delivery method (such as online, face-to-face, blended, immersive, etc.) for a set of lessons to teach the Blackfoot language to Blackfoot children will have ramifications for the design that were probably not even considered when the original problem was outlined. Deciding to use a traditional story as a computer game, as was done in one situation (Parker, Heavy Head, & Becker, 2005) created a whole new set of problems that would not have been there if the decision had been to use a live in-class approach.

9. A number of different stakeholders are interested in how it is solved.

This may well be the trademark characteristic of wicked problems. Aside from the learners and the teacher or facilitator, there are often many other stakeholders that are involved either directly or indirectly: parents, administrators, funding agencies, government agencies, accreditation bodies, professional organizations, etc. The existence of a discrepancy in representing a Wicked Problem can be explained in numerous ways. One is that the choice of explanation determines the problem's resolution (Lewandowski, Gutschow, McCartney, Sanders, & Shinners-Kennedy, 2005). Since we are directly affected by various stakeholders as well as the context and constraints of the problem, something like the relative weighting of the various elements of the design model used can dramatically affect the directions taken during the solution. Each stakeholder brings a distinct set of values and experiences to the problem, which affects the judgments about the nature of the problems as well as the value of a solution. For example, if the stakeholders determine that the learner characteristics are *the* key element this can in turn determine the choices of venue.

10. The planner has no right to be wrong.

A poor design in ID almost always has lasting repercussions. The whole point of instructional design is to create an intervention to teach someone something. The solution will be implemented within a social context (on one or more people), so people will be directly affected. Since ID solutions cannot be tested without being implemented on real people, the side-effects of bad choices can be costly. The designer typically does not ethically have the freedom to postulate a solution 'just to see what happens', or worse, attempt one that may have predictable negative consequences. There will at the very

least be learners who are affected by our solution, and there is no way to really test our solution without trying it out on someone, so the risks are always high. Similarly, a request to 'strike it from the record', or a direction to the learners to 'disregard that last lesson' cannot erase what has been done. We may have failed in meeting our stated instructional objectives, which is not the same as being able to claim that nothing happened. What Rittel and Webber said of social planning problems can also be applied to instructional design: "We have neither a theory that can locate societal goodness, nor one that might dispel wickedness, nor one that might resolve the problems of equity that rising pluralism is provoking" (Rittel & Webber, 1973, p. 169) . The issues raised here are not likely to be solved, but recognizing the complexity of the problems can suggest approaches that remain sensitive to them.

Approaches to Wicked Problems

There is reason to believe that many instructional designers spend approximately half their time involved in either original design activities or project management (Cox & Osguthorpe, 2003) so an approach that addresses both activities simultaneously seems promising. If we accept that ID is a wicked problem, the next obstacle to tackle is how to approach a solution. The problems are real and solutions are necessary. Knowing that a problem is wicked is of no use if that does not also help us to address it.

A first step is to remove oneself of the bindings associated with any one particular model and sometimes the theories that underpin it, but without abandoning the valuable guidance each can provide—flexibility is key in the solution of a wicked problem. One way to achieve this is to notice that while each of the ID models presented may appear quite distinct as shown, but when they are viewed from a higher level of abstraction, it is found that underlying each approach are many commonalities. The result of this unravelling of other models is a 'meta-model' such as the one shown below. This meta-model essentially flattens the hierarchy of the other models, thus freeing the implementer to create his or her own specific, relevant, and informed hierarchy of priorities and relationships. This allows us the flexibility needed to cope with the capricious nature of wicked problems.

Each of the other models proposes various sub-problems to be solved and asks various questions that require answers. When taken all together, *these questions* make up the set of queries, which are collectively called the 'Guiding Questions' here. In a 'Wicked ID' approach, these 'Guiding Questions' are all initially assigned the same weighting in terms of relative importance. The questions are used to drive the design process rather than a pre-defined set of steps or tasks. Allowing the questions themselves to drive the process causes them to be generated, categorized and prioritized to meet the needs of the specific instructional problem(s) being addressed.

Upon repeated use, it will likely be found that a general pattern of organization emerges in a particular organization, with particular design teams, or in response to particular kinds of performance problems. This encompasses the core ideas behind the concept of *"patterns"*as defined in software design (Fitzpatrick, 2003), which can just as effectively

be applied to instructional design (Brophy & Velankar, 2006).

Model Process

No perspective on instructional design would be complete without at least some reference to the *process* to be employed, regardless of how flexible and plastic it must to be. Without at least some structure, progress towards a solution cannot be made. Here, the approaches are divided into those typically employed at the macro level of design (addressing the performance problem), and those applied at the micro level (tasks). The ultimate design goal at the macro level of design is consensus, which can be achieved through discussion of analyses, alternate solutions, understandings, contexts, priorities, and assessments. All processes must be adaptive in order to lead to a satisfactory resolution. One approach known as Scrum (Gamma, 1995) has been used successfully in new product development projects, and has more recently been applied to computer software development, as well as Computer Supported Cooperative Work (CSCW) (DeGrace & Stahl, 1990)with reasonable success. It could be argued that any instructional design solution qualifies as a new product and thus may also benefit from this approach.

A Scrum project collects input from all stakeholders, into a list called the 'Backlog'. In our case, this input will come as a result of an examination and ordering of the guiding questions—some may be presented in the form of a 'wish list', others will remain as questions that need answers, others still will be identified as essential elements that must be implemented. In each time period, known as a 'Sprint' (usually one month), the development team selects as many of the top priority features from the Backlog as it can develop in that time. After the time is up, the results are presented to the stakeholders, who provide feedback and may reorganize, modify or even add to the Backlog. This means that new questions, tasks, or other elements can be added after each sprint. Then the team selects a new set. Each feature is associated with a time-line or time limit (and other required resources). This gives the project leaders a means of tracking progress. The entire Backlog estimates are graphed against the actual cost every three or so rounds. This gives a reliable indication of whether the project is converging towards or diverging from an acceptable solution.

When constraints or the situation dictates that a small design team must create the solution, several other processes are possible. Handcuffing is often employed when working in small groups. This process effectively 'ties' a domain expert to a single designer to provide feedback and serve the role of formative evaluator for the duration of the project. This can also be made to work when there are more than two, but typically less than five people on the design team. When no domain experts are available (other than the requisite sources for data gathering), and the design team consists of one or two individuals, the process is called 'hacking'.

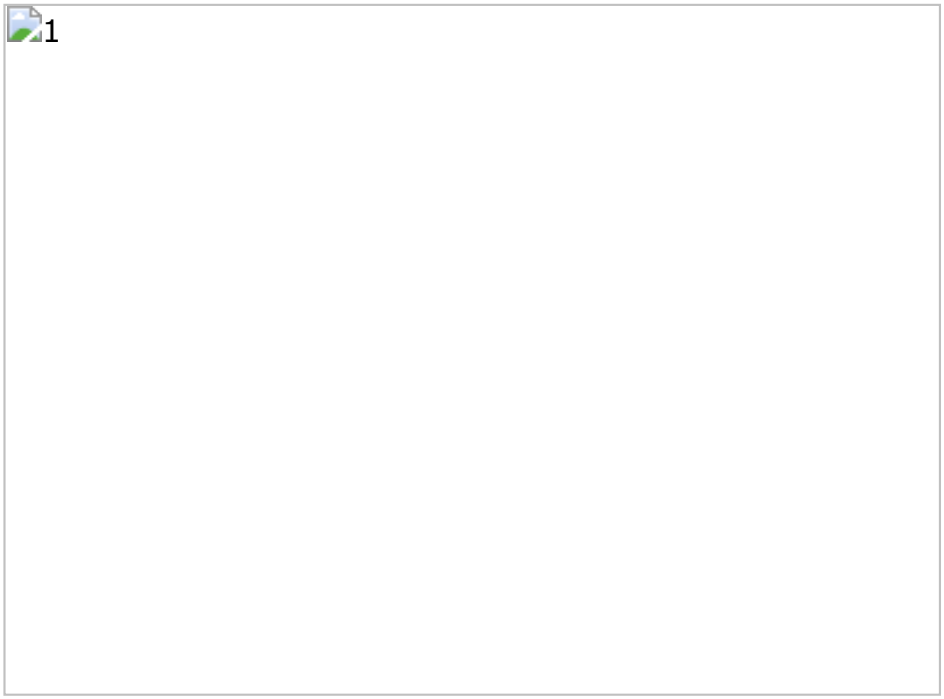
While numerous other methodologies are currently used in lightweig ht, agile software development, most cannot be effectively applied to instructional design. ID bears many similarities to software design problems but it has one key distinction: it has the added

complication that interim or preliminary testing or prototyping (as defined and practiced in SENG) is only possible if the client accepts that the 'debut' of any instructional solution developed this way will inevitably require (possibly major) adjustments. Many other effective software design methodologies, such as Rapid Prototyping, Cleanroom Approaches, Extreme Programming, the Sashimi methodology, (Fitzpatrick, 2003) include early production of an incomplete working model that will be tested and adjusted in subsequent iterations. It can be argued that this strategy simply cannot be applied when the object of the design is effective instruction of human subjects in the upper levels of the cognitive domain (analysis, synthesis, evaluation). Finally, when it comes to assessment of 'functionality', comparisons between software and instruction break down.

Using the Model

Macro Application

The development process begins with a first pass through the guiding questions, which will be categorized according to a scheme agreed upon by the development team. At the very least the questions should be grouped into those that will be addressed and those that are considered irrelevant in this application. The visual representation of the model shows five categories for questions kept, and the percentages are intended to provide a rough guideline for relative importance. For example, given the question, "What is the facilitator's formal training?", how important is it that question be addressed? If it is something that will significantly affect the design, or something where the answer is non-negotiable, then it should be placed at a fairly high priority. This question may in turn inspire other questions or limitations that need to be stated.



Once we have completed a first pass through the Guiding Questions, we are ready to begin the Scrum process. At this point we should have a general direction and an overview of the instructional goals. Going through all the questions as a group helps to put everyone 'on the same page', and even though some disagreement is likely, members will at least know

which issues are likely to be straightforward, which will be contentious, and which will require substantial support to address. This process may take several rounds, but the ultimate goal at this level is consensus (meaning permission to proceed) rather than total agreement. This process is highly collaborative, and if serious disagreement has arisen that appears insoluble, it may be necessary to change to a more structured, hierarchical development process. The 'final' list may also include a list of concepts, topics, or skills to be mastered, but the details may still be somewhat nebulous. In essence, after the initial stages, the macro level application may begin to loose focus and the micro level implementations take over. Given the nature of the process however, the macro level must remain in some degree of focus as a change always has the potential to alter our direction. It is accepted that the 'final' list is not necessarily final until the entire process is complete. The process is complete when we agree that what we have is 'good enough', or we run out of resources.

Micro Application

The ultimate goal at the micro level of application is efficient and effective learning (DeGrace & Stahl, 1990). Here the learners and facilitators are often the primary stakeholders. The usual aim is to apply the Guiding Questions at the micro level and end up with a series of smaller units that can now be identified as tame, whereupon the designers can revert to other well-known methodologies (Kopp, personal communication, 2002). If an element cannot be tamed, then the same approach (Scrum) can be applied at this level too.

Example Design

The following paragraphs outline some of the early phases of one instructional design project. In this case, the task is to design an introductory fast-track programming course for entry-level computer science students enrolled in a degree program at a university. We were able to assume some prior learner programming experience [*Q: We need to settle on how much we will assume.*], and we are told that the learners are ranked among the top 10% of their peers. We have an upper limit of 40 on the class size and a lower limit of 25 [*Q: What will happen to registered students if we don't meet the minimum numbers?*]. We've been told that we are to use an object-oriented programming language [*Q: Who decides which one?*]. There will be a single instructor who will also be responsible for 'teaching' labs. The initial proposal calls for 3 lecture and 4 lab hours per week over a typical 13-week course.

At some point in the initial process, the designer(s) will determine that enough questions have been generated along with some potential solutions. Unlike many other ID models, brainstorming processes that generate specific instructional strategies, materials, or sequences are encouraged even in the initial stages, but no decisions are made about implementation at this point. Ideas are noted, and held over until later. Some questions will probably have been asked that will need to be researched. This provides us with a set of Guiding Questions specifically tailored to this problem. The designers will then meet with the stakeholders who will provide some answers as well as new input. In this example the

stakeholders would likely include the Curriculum Committee, the Head of Department, the Instructor, and possibly the Dean of the Faculty. Other faculty who have taught first year courses in the past may also be involved.

Suppose the answer to the question of how much prior programming experience is to be required of students turns out to be split with some willing to accept very little and others expecting the equivalent of a half-course. The consequences of various approaches on just this one issue can be explored until we reach a consensus. In this example, let's assume the consensus reached involves the design and application of an initial assessment instrument for the students, and a subsequent evaluation just before the deadline to change classes. It is assumed that students who do not pass the subsequent evaluation will be advised to switch into the "regular" class. Students can be made aware of this requirement in advance of the course.

Eventually, the general content and syllabus are outlined and again presented to the stakeholders, whereupon they may decide to retain some portions, modify others, and delete others still. However, since the nature of this process is not linear, a less likely, but still possible outcome would be that a more-or-less complete potential solution is presented and accepted early on. When this happens, the problem has been tamed and implementation can follow.

While it may be true that this sort of process is sometimes followed in practice, it is still the case that major revisions to a design are typically viewed as failures rather than progress towards an eventual resolution. Acceptance of this process and negotiations during it may go more smoothly if it is realized from the outset that "going back to the drawing board" is simply another part of the process, and presentations of solutions are not to be treated as final.

Reflections

Strengths

The process itself is highly adaptable. The guiding questions can be added to as necessary —the initial set is exactly that: *initial.* They are intended as a set of questions to begin the process—the final set may have only a few of the original questions, but a large number of questions added during the organizational meetings. It must be recognized that the very nature of wicked problem solution involves a fluid, interconnected approach, and so the initial question set will always be subject to change and revision.

Weaknesses

Given that the approach is inherently unstructured, it does not attempt to provide a road map for solving any particular instructional problem. It assumes a design team that includes at least one individual with extensive knowledge of existing and well-established design strategies, as it must be recognized that this approach canno t be used in isolation from other models. In other words, this model provides a rough framework within which other models can be applied as loosely as necessary.

Limitations

This approach assumes a basic level of design competence. It is intended to help solve complex problems, and could be described as overly complex and unstructured for use with smaller, clear cut problems. Once a problem or portion thereof has been tamed, other approaches become more effective than the one outlined here. The "all at once" strategies such as Scrum, Handcuffing, and Hacking assume all people involved in the design have similar or at least complimentary goals and values (honesty, integrity, ethics). Each effort must begin with a consensus of the fundamental goals, as such an open-ended approach is unlikely to succeed if one or more parties carry hidden agendas.

Conclusions

Perhaps the biggest benefit of the recognition of ID as a wicked problem is that it is liberating. It frees us to think about the problems and process in new ways. It allows us to look to other venues for inspiration, while still not renouncing the value of known and tested models and approaches, but rather adding to them. Perhaps an analogy to end this paper would be fitting. Bruce Lee, the world-renowned martial artist, faced great controversy in his last years. The main reason was that he was using and applying various martial arts in novel and 'un-approved' ways. The result was the development of an entirely new martial art, called Jun Fan Jeet Kune Do (Lee & Little, 1997). Many of the martial arts have very long histories and their philosophies, styles, and moves have been perfected over time to a very precise art and science. Bruce Lee's genius was in choosing from among many formal martial arts and applying them to suit the situation. The reason it worked so well in his case, was that he also possessed the skill, expertise, and dedication to master each of the other arts that he used. A prerequisite for entry into many of the training schools that now teach Bruce Lee's method remains a background in other martial arts. Similarly, a prerequisite for the use of approaches to ID as a wicked problem might well include a thorough knowledge of other ID models.[2]

1 A **stopping rule** is characterized as a mechanism for deciding whether to continue or stop a process on the basis of the present position and past events, and which will almost always lead to a decision to stop at some time. [source: Wikipedia:http://en.wikipedia.org/wiki/Stopping_rule retrieved: Mar 13 2006] For example, if my problem is opening a jar, then the stopping rule would be stop turning the lid when it comes off.

2 Consensus, implying permission to proceed, rather than agreement.

References

Beck, K. (1999). Embracing change with extreme programming. *Computer Science Education, 32*(10), 70–77.

Beck, K. (2000). *Extreme programming explained: embrace change*. Reading, MA., Harlow, England: Addison-Wesley.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., et al. (2001). *Manifesto for agile software development*. Retrieved March 2005 fromhttp://agilemanifesto.org/

Bereiter, C. (2002). Design research for sustained innovation. *Cognitive Studies, Bulletin of the Japanese Cognitive Sceince Society, 9*(3), 321–327.

Bichelmeyer, B., Boling, E., & Gibbons, A. S. (2006). Instructional design and technology models: Their impact on research and teaching in instructional design and technology. In M. Orey, V. J. McClendon & R. M. Branch (Eds.), *Educational media and technology yearbook*(Vol. 31, pp. 33–73). Littleton, CO.: Libraries Unlimited, Inc.

Brooks, F. P. (1975). *The mythical man-month: essays on software engineering*. Reading, MA.: Addison-Wesley.

Brophy, S., & Velankar, Y. (2006). *Work in progress: Cataloging instructional design patterns that facilitate generative learning*. Paper presented at the Frontiers in Education Conference, 2000. FIE 2000. 30th Annual FIE'06 Proceedings 36th Annual Conference.

Budgen, D. (2003). *Software design* (2nd ed.). New York: Addison-Wesley.

Conklin, E. J., & Weil, W. (1997). Wicked problems: naming the pain in organizations. Retrieved October 18 2002 fromhttp://www.mmm.com/meetingnetwork/readingroom/gdss_wicked.html

Cox, S., & Osguthorpe, R. T. (2003). How do instructional design professionals spend their time? *Tech Trends, 47*(3), 45–47.

DeGrace, P., & Stahl, L. H. (1990). *Wicked problems, righteous solutions: A catalogue of modern software engineering paradigms*. Englewood Cliffs, N.J.: Yourdon Press.

Derry, S. J., & Steinkuehler, C. A. (2003). Cognitive and situative theories of learning and instruction. In L. Nadel (Ed.), *Encyclopedia of Cognitive Science* (pp. 800–805). England: Nature Publishing Group.

Dick, W., Carey, L., & Carey, J. O. (2001). *The systematic design of instruction* (5th ed.). New York: Longman.

Doll, W. E. (1993). *A post-modern perspective on curriculum*. New York: Teachers College Press.

Donald, M. (2001). *A mind so rare: The evolution of human consciousness* (1st ed.). New York: W.W. Norton.

Douglas, I. (2006). Issues in software engineering of relevance to instructional design.*TechTrends, 50*(5), 28–35.

Fitzpatrick, G. (2003). *The locales framework: understanding and designing for wicked*

problems: Kluwer Academic Publications.

Ford, G., & Gibbs, N. E. (1996). *A mature profession of software engineering*. Pittsburgh, PA: Carnegie Mellon University.

Gamma, E. (1995). *Design patterns: Elements of reusable object-oriented software*. Reading, MA.: Addison-Wesley.

Gerlach, V. S., & Ely, D. P. (1980). *Teaching and media: A systematic approach* (second ed.). Englewood Cliffs, N.J.: Prentice Hall, Inc.

Gibbons, A. S. (2003). What and how do designers design? A theory of design structure.*Tech Trends, 47*(5).

Hannafin, M. J., & Peck, K. L. (1988). *The design, development, and evaluation of instructional software*. New York, London: Macmillan Collier Macmillan.

Jonassen, D. H. (1997). Instructional design models for well-structured and ill-structured problem-solving learning outcomes. *Educational technology research and development: ETR & D, 45*(1), 65 (30 pages).

Jonassen, D. H. (2004). *Learning to solve problems: An instructional design guide*. San Francisco, CA: Pfeiffer.

Jonassen, D. H., Cernusca, D., & Ionas, G. (2006). Constructivism and instructional design: The emergence of the learning sciences and design research. In *Trends and issues in instructional design and technology* (2nd ed., pp. 45–61). Upper Saddle River, N.J.: Merrill/Prentice Hall.

Jonassen, D. H., & Rohrer-Murphy, L. (1999). Activity theory as a framework for designing constructivist learning environments. *Educational Technology Research and Development, 47*(1), 61–79.

Kenny, R. F., Zhang, Z., Schwier, R. A., & Campbell, K. (2005). A review of what instructional designers do: Questions answered and questions not asked. *Canadian Journal of Learning and Technology, 31*(1), 9–26.

Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J.-M., et al. (1997). *Aspect-oriented programming.* Paper presented at the European Conference on Object-Oriented Programming.

Kopp, G. (2002). Personal communication: Faculty of Education, University of Calgary.

Krill, P. (2006). Agile programming has fallen short, conference told [Electronic Version].*InfoWorld*. Retrieved March 16 2006 fromhttp://www.infoworld.com/article/06/03/13/76420_HNmcconnell_1.html.

Kruse, R. L. (1982). On teaching recursion. *ACM SIGCSE Bulletin , Proceedings of the*

thirteenth SIGCSE technical symposium on Computer science education SIGCSE '82, 14(1), 92–96.

Lakoff, G. (1987). *Women, fire, and dangerous things: What categories reveal about the mind*. Chicago: University of Chicago Press.

Lee, B., & Little, J. R. (1997). *The tao of gung fu: A study in the way of Chinese martial art*. Boston: C.E. Tuttle.

Lewandowski, G., Gutschow, A., McCartney, R., Sanders, K., & Shinners-Kennedy, D. (2005, October 2005). *What novice programmers don't know.* Paper presented at the Proceedings of the 2005 international workshop on Computing education research ICER '05.

Meltzer, D. E. (2004). How do you hit a moving target? Addressing the dynamics of students' thinking. In J. Marx, P. Heron & S. Franklin (Eds.), *CP790, Physics Education Research Conference*: American Institute of Physics.

Molenda, M. (2003). In search of the elusive ADDIE model. *Performance Improvement*.

Morrison, G. R., Ross, S. M., & Kemp, J. E. (2004). *Designing effective instruction* (4th ed.). Hoboken, N: J.: Wiley & Sons.

Nelson, H. G., & Stolterman, E. (2003). *The design way: Intentional change in an unpredictable world: Foundations and fundamentals of design competence*. Englewood Cliffs, N.J.: Educational Technology Publications.

Parham, J. (2003). Assessment and evaluation of computer science education. *Journal of Computing Sciences in Colleges, 19*(2), 115–127.

Parker, J. R., Heavy Head, R., & Becker, K. (2005, October 13–15 2005). *Technical aspects of a system for teaching aboriginal languages using a Game Boy.* Paper presented at the Future Play, The International Conference on the Future of Game Design and Technology, Michigan State University, East Lansing, Michigan.

Poppendieck, M. (2004, April 12 2004). Wicked projects. Retrieved September 1, 2004 fromhttp://poppendieck.com/wicked.htm

Poppendieck, M., & Poppendieck, T. D. (2003). *Lean software development: An agile toolkit*. Boston: Addison-Wesley.

Rittel, H. W. J., & Webber, M. M. (1973). Dilemmas in general theory of planning. *Policy Sciences, 4*, pp155–169.

Rittel, H. W. J., & Webber, M. M. (1984). Planning problems are wicked problems. In N. Cross (Ed.), *Developments in design methodology* (pp. 135–144). New York: John Wiley and Sons.

Rothwell, W. J., & Kazanas, H. C. (1998). *Mastering the instructional design process: A*

systematic approach (2nd ed.). San Francisco, CA.: Jossey-Bass.

Schön, D. A. (1983). *The reflective practitioner: How professionals think in action*. New York: Basic Books.

Schrage, M. (2000). *Serious play: How the world's best companies simulate to innovate*. Boston: Harvard Business School Press.

Schwier, R. A., Hill, J., Wager, W., & Spector, J. M. (2006). Where have we been and where are we going? Limiting and liberating forces in IDT. In M. Orey, V. J. McClendon & R. M. Branch (Eds.), *Educational media and technology yearbook* (Vol. 31, pp. 75–95). Littleton, CO.: Libraries Unlimited, Inc.

Seels, B., & Richey, R. (1994). *Instructional technology: The definition and domains of the field*. Washington, D.C.: Association for Educational Communications and Technology.

Simon, H. A. (1973). The structure of ill structured problems. *Artificial Intelligence, 4*(3), 181–201.

Stewart, I., & Cohen, J. (1997). *Figments of reality: The evolution of the curious mind*. Cambridge ; New York: Cambridge University Press.

Tomasello, M. (1999). *The cultural origins of human cognition*. Cambridge, Mass.: Harvard University Press.

Tripp, S. D., & Bichelmeyer, B. (1990). Rapid protoyping: An alternative instructional design strategy. *Educational Technology, Research and Development, 38*(1), 31–44.

Varela, F. J., Thompson, E., & Rosch, E. (1991). *The embodied mind : cognitive science and human experience*. Cambridge, MA.: MIT Press.

Wang, W.-L. (2002). Beware the Engineering Metaphor. *Communications of the ACM, 45*(5), 27–29.

Weinberg, G. M. (1998). *The psychology of computer programming* (Silver anniversary ed.). New York: Dorset House Pub.

Wiggins, G. P., & McTighe, J. (1998). *Understanding by design*. Alexandria, VA.: Association for Supervision and Curriculum Development.

Willis, J. (2000). The maturing of constructivist instructional design: Some basic principles that can guide practice. *Educational Technology, 40*(1), 5–16.

ISSN: 1499-6685